AMENDMENTS TO THE DRAWINGS

The attached sheet of drawings includes changes to Fig 2.

Attachment: Replacement sheet

                   Annotated sheet showing changes

REMARKS

The official action provisionally rejects claims 1, 2, and 4 – 19 on the grounds of nonstatutory obviousness-type double patenting in light of claims 1 – 26 of copending Application Serial No. 10/395,557. The action rejects claims 21 and 22 under 35 U.S.C. § 112, ¶ 2, as being indefinite. The action rejects claims 1, 2, and 4– 22 under 35 U.S.C. § 102 as anticipated by U.S. Patent No. 6,397,242 ("*Devine et al.*").

Claims 1, 2, and 4-22 were pending as of the last office action. Claim 2 has been canceled, without prejudice to re-instatement, by amendment above. Claims 1, 4, 7, 11, 17, and 18 have been amended to include the subject matter of (or similar to) claim 2.

1.   PROVISIONAL OBVIOUSNESS TYPE DOUBLE PATENTING REJECTION

Applicant acknowledges the provisional double patenting rejection of claims 1, 2, and 4 – 19 in view of copending Application Serial No. 10/395,557. Because the rejection is provisional, applicant takes no action at this time in regards to this rejection.

2.   REJECTION UNDER 35 U.S.C. § 112, ¶ 2

Applicant respectfully traverses the rejections of claims 21 and 22. Applicant has amended each claim to refer to "end of instructions quota." Paragraph 0032 describes an example of what is meant by the term. As noted at paragraphs 0027 and 0028 a direct execution monitor in accordance with the present disclosure may be able to manage data transfer and instruction execution between virtual machines. With this functionality, for example, guest code may run on multiple simulated virtual machines, each simulating a processor. The direct execution monitor manages transfers among these virtual machines, such as with a hyper-threading system. In any event, paragraph 0032 describes that the direct execution monitor (e.g., a block 308) may initialize the simulation context by assigning a particular quota of instructions to be executed by each simulated CPU, i.e., by each virtual machine. The direct execution monitor then can determine if a virtualization event has occurred where one virtual machine has reached its allotted instruction quota, in which case the direct execution monitor may switch control to another virtual machine for further instruction execution.

Applicant respectfully asserts that persons of ordinary skill in the art will understand the meaning of each of terms used in the claims.

The rejection is traversed.

8

4. PRIOR ART CLAIM REJECTIONS

Applicant has carefully considered the office actions rejections and respectfully traverses.

Devine et al. relates to a virtualization system that employs a virtual machine monitor (VMM) on segmented-architecture computers. That VMM comprises three subsystems: a binary translation subsystem; a direct execution subsystem; and a decision module/subsystem that discriminates between code that is executable in the preferred direct execution subsystem and code that is only executable in the binary translation subsystem. See, e.g., *Devine et al.* 5:20-31 and *Devine et al.* Figure 8. As *Devine et al.* describes, virtual machines (VMs) would preferably execute in a direct execution environment, in which the virtual machine runs directly on the hardware to maximize performance. *Devine et al.* 4:47-54. *Devine et al.* recognize, however, that complete virtualization in a direct execution environment is not achievable with its system, which is why *Devine et al.* use a VMM that also includes a binary translation subsystem for handling non-virtualizable instructions not directly executable in the VM. *Devine et al.* 5:20-31.

Devine et al. distinguishes its system-level VMM (which does not exist on the host operating system) from VMM that operate on top of an operating system. *Devine et al.* defines the VMM of its disclosure as a "thin piece of software that runs directly on top of the hardware and virtualizes all the resources of the machine." *Devine et al.* 1:44-46. That is the VMM of *Devine et al.* does not operate on top of or in the Host operating system, as is the case for the direct execution monitor of FIGS. 2 and 3 for the present application. In fact, *Devine et al.* expressly states that virtual machine monitors operating on top of a host operating system are completely different than the VMM described in their work:

> Although these systems are commonly referred to as a form of **virtual machine monitor, they run either on top of an existing operating system**, such as DOSEMU, or as part of an existing operating system such as Microsoft Windows and Microsoft NT. In this respect, **they are quite different from the true virtual machine monitors described above, and** from the definition of the term 'virtual machine monitor' **applied to the invention described below**." *Devine et al.* 3:65-4:5

Further quotations regarding the system level implementation of the VMM are also provided:

> Conventional VMMs outperform machine emulators since they run at system level without the overhead and constraint of an existing operating system. 4:24-26.

> FIG. 7 shows the total system incorporating the invention
> in its broadest terms: a protected host operating system (HOS)
> 700 is combined with at least one unconstrained, system-level
> virtual machine monitor (VMM) 100 according to this invention.
> 24:8-12.

In other words, the VMM of *Devine et al.* does not operate in or on top of the host operating system. Therefore, the VMM of *Devine et al.* cannot be said to teach or suggest providing "a single monitor <u>within the host operating system environment</u> to control entry to and exit from each of the plurality of virtual machines in the direct execution environment," as recited in claim 1. For this reason alone, the rejection of claim 1 is traversed.

Furthermore, Applicant re-asserts that *Devine et al.* does not appear to teach a single monitor that controls entries and exists to a plurality of virtual machines. Applicant previously noted that, at Column 24, lines 18-26, *Devine et al.* describes that each VMM is limited to supporting a single VM, such that if multiple VMs are desired then multiple VMMs must be employed:

> FIG. 7 shows one virtual machine monitor 100 supporting one
> virtual machine 120. The system according to the invention
> makes it possible to include any number of VMM's in a given
> implementation, **each supporting a corresponding VM**, limited
> only by available memory and speed requirements, and to switch
> between **the various included VMM's**. It is assumed merely for
> the sake of simplicity that the VMM 100 is the one actively in
> operation on the system. *Devine et al.* 24:18 – 26.

This requirement for a separate VMM for each VM also appears to be described in the multiple processor emulation application, where *Devine et al.* explains that separate global and local shadow descriptors (one for each VMM) are required for each virtualized processor, thus preventing one monitor from handing multiple processor emulations:

> The use of multiple processors allows, for example, the
> simultaneous execution of multiple virtual machines, or the
> execution of a virtual machine with multiple virtual processors.
> **The virtualization of each processor is handled separately
> and independently**, including the decision as to whether to use
> direct execution or binary translation. For each virtual processor,
> the VMM will then maintain a separate set of global and local
> shadow descriptor entries. *Devine et al.* 25:7 – 14.

The Examiner responds to these descriptions by citing three portions of *Devine et al.* **First**, the Examiner cites to Column 1, lines 53-64. Applicant responds that this section is in the

background section of the specification and is not described as necessarily speaking to functionality of the VMM used in the *Devine et al.* system. Furthermore, this section of *Devine et al.* does not state that a single virtual machine was used to control entry and exits to multiple VMs running in a direct execution environment. **Second**, the Examiner cites to Column 11, lines 34-48. Here *Devine et al.* is discussing memory tracing, i.e., the ability of the VMM to set read-traces or write-traces on a given physical page of a virtual machine. The section refers to a specific feature of conventional virtual machine monitors, i.e., containing a mechanism that virtualizes the physical address space of a virtual machine. There is mention that conventional virtual machine monitors support multiple virtual machines, but there is no teaching of providing "a single monitor within the host operating system environment to control entry to and exit from each of the plurality of virtual machines in the direct execution environment," as recited in claim 1. **Third**, the Examiner cites to Column 25, lines 7-21, and argues that the Applicant has only cited to a portion of this section. Applicant notes that the office action nowhere specifies where it believes certain claimed subject matter is taught in the prior art. The office action cites lines of text in *Devine et al.*, but many times does so without explaining the relevance of any of the cited language. The citation to Column 25 is an example. There is nothing in this section that teaches "a single monitor within the host operating system environment to control entry to and exit from each of the plurality of virtual machines in the direct execution environment." This entire section is entitled "Multi-Processor Embodiment" and describes that the hardware 710 may include more than one <u>physical</u> processor. It is this use of multiple processors that would allow the execution of "multiple virtual machines, or the execution of a virtual machine with multiple virtual processors." The only modifications to the MMU described in this example pertain to having some interaction between the physical processors when the memory traces are installed and when traced access occurs. There is no teaching or suggestion of "provide a single monitor within the host operating system environment to control entry to and exit from each of the plurality of virtual machines in the direct execution environment," as recited in claim 1.

Applicant traverses the rejection of claim 1 on these grounds as well.

The rejection of claim 1 and the claims depending therefrom are respectfully traversed.

In addition to being in condition for allowance by implication, applicant separately highlights some of the claims depending from claim 1, in part because they independently recite allowable subject matter and in part because the office action's rejection of these claims is unclear. Dependent claim 4 generally relates to controlling transfer of virtual code between a host environment and a virtual environment based on a virtualization event attempted by a

11

virtual machine. Whereas the present application describes a direct execution monitor that runs **on top of or in a Host operating system** and in some instances can execute virtualization events or in other instances de-virtualize those events and pass them to the full platform simulator for execution, *Devine et al.* teaches that non-virtualizable events are passed to the binary translation subsystem of a VMM operating at **the system level directly over hardware**. There is no teaching of passing such events to the Host operating system for execution. Dependent claims 5-6 describe further details also not taught by *Devine et al.*

     Devine et al. cannot be said to teach any of the recitations of dependent claims 4-6.

     Claim 21 depends from claim 1 as well (indirectly through claim 20) and generally relates to determining if the virtualization event is an end of instructions quota event, reflecting the end of instructions to be executed on that virtual machine, in response to which the system switches instruction execution to another one of the plurality of virtual machines. The Examiner points to Column 11, line 45 – Column 12, line 67 as teaching transfer of instruction execution from one virtual machine to another virtual machine in response to an end of quota event. Applicant respectfully disagrees. This portion of *Devine et al.* does describe a MMU, but nowhere has the Examiner identified (nor can the applicant's representative spot) any teaching that the MMU or VMM passes control from one virtual machine to another in response to an end of instructions quota event in one virtual machine. As noted above, it is questionable whether *Devine et al.* even teaches that its single VMM controls access to and from multiple VMs. Either way, though, there is no teaching of how the system would pass control between VMs. The office action cites to descriptions of three core modules of the VMM that use memory tracing: 1) virtualizing segmented architecture of the virtual processor; (2) virtualizing page table based MMU of the virtual processor, to guarantee coherency of the shadow copy; and (3) to guarantee coherency of the translation cache. Memory tracing techniques for transparent read and write traces are described, but nothing teaches determining "if the virtualization event is an end of instructions quota event for one of the plurality of virtual machines" or "in response to an identification of the end of instructions quota event, [switching] to another one of the plurality of virtual machines," as recited in claim 21.

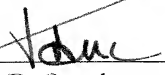   *Devine et al.* cannot be said to teach claim 21 as a result.

     For at least the reasons outlined above with respect to claim 1 (and claims 4-6 and 21), claim 11 (and the claims depending therefrom) is neither anticipated nor rendered obvious by the

prior art. The rejection of claim 11 is traversed. Claim 11 and claims 12 – 16 and claim 22 depending therefrom are in condition for immediate allowance.

For at least the reasons outlined above – namely that *Devine et al.* does not teach or suggest a single monitor running in host operating system and being capable of controlling multiple virtual machines in a direct execution environment – claim 17 is neither anticipated nor rendered obvious by the prior art. The rejection of claim 17 is traversed. Claim 17 and claims 18 and 19 depending therefrom are in condition for immediate allowance.

In view of the above amendment, applicant believes the pending application is in condition for allowance.

Dated: March 7, 2007

Respectfully submitted,

By _____

Paul B. Stephens
   Registration No.: 47,970
MARSHALL, GERSTEIN & BORUN LLP
233 S. Wacker Drive, Suite 6300
Sears Tower
Chicago, Illinois 60606-6357
(312) 474-6300
Attorney for Applicant

Attachments